ISSN 1799-2591
Theory and Practice in Language Studies, Vol. 11, No. 9, pp. 1041-1049, September 2021
DOI: http://dx.doi.org/10.17507/tpls.1109.09

The Application of NLTK Library for Python Natural Language Processing in Corpus Research*

Meng Wang
School of Medical Information Engineering, Jining Medical University, Rizhao, Shandong Province, China

Fanghui Hu
School of Foreign Languages, Jining Medical University, Rizhao, Shandong Province, China

***Abstract*—Corpora play an important role in linguistics research and foreign language teaching. At present, the relevant research on the corpus in China mainly uses WordSmith, Antconc and other retrieval tools. NLTK library, which is based on Python language, can provide more flexible and rich research methods, and it can use unified data standards to avoid the trouble of various data type conversion. At the same time, with the help of Python's numerous third-party libraries, it can make up for the shortcomings of other tools in syntax analysis, graphic rendering, regular expression retrieval and other aspects. In terms of the main links in corpus research, such as text cleaning, word form restoration, part of speech tagging and text retrieval statistics, this paper takes the US presidential inaugural speech in the corpus as an example to show how to use this tool to process the language data, and introduces the application of Python NLTK library in corpus research.**

***Index Terms*—corpus, python, natural language processing, NLTK**

I. Introduction

At present, many fields of linguistic research pay more attention to the application of the corpus, for the corpus, taking massive and real language data as the research objects, is scientific and accurate (Feng, 2020). With the rapid development of computer technology, the corpus has entered a stage of systematic theoretical innovation and extensive application in the field of linguistics. More and more researchers from different academic backgrounds have joined the corpus research team, and many research fields, such as lexicography, sociolinguistics, stylistic analysis, pragmatics and so on, cannot do without corpus.

Both the construction of the corpus and the study of the corpus are inseparable from the processing of corpus data. Currently, the commonly used corpus processing tools include WordSmith, AntConc, Range, PowerGREP, etc. Most of the above tools provide functions such as retrieval, segmentation, substitution, statistics, etc. However, they are limited to the lexical and lexical collocation level, rather than the syntactic and discoursal level. In addition, due to the limitation of the software design, it cannot be flexibly customized, so researchers may have to learn the operation of different software when necessary. The NLTK library, based on the computer programming language Python, is a toolkit that can be used for natural language processing. The toolkit not only has the retrieval function commonly seen in the above tools, but also has many functions such as text cleaning, word form merge, part of speech tagging, grammar analysis and semantic analysis. Through this toolkit, researchers can complete the whole process from corpus construction to research retrieval in one environment, eliminating the inconvenience of switching between different software and data conversion, and further expanding the scope and depth of research.

Based on the introduction to the application of Python NLTK library in terms of text cleaning, word form restoration, part of speech tagging and text retrieval statistics, this paper will take the American presidential inaugural speech in the corpus as an example to introduce how to use the natural language processing toolkit to process the language data. Consequently, corpus researchers will get familiar with and use them, research tools will increase, and corpus linguistics research will develop rapidly.

II. Introducton and Installation of NLTK Library

NLP (Natural language processing) is a science integrating linguistics, computer science and mathematics. The research in this field will involve natural language (that is, the language used by people in daily life), so it is closely related to the research of linguistics, and is an important direction in the field of computer science and human

*This paper is supported by Education research project of Jining Medical University in 2018: construction of micro-course resource library of Fundamentals of Program Design for medical students (No. : 18049)

intelligence. It mainly studies various theories and methods that can realize effective communication between human and computer with natural language. At present, the main computer programming language used in natural language processing is python.

As a high-level programming language, Python, with its elegant, concise, clear rules, is very suitable for non-computer professionals to learn. In addition, Python has a large number of third-party expansion library support, making it greatly applied in the web crawler, data analysis, machine learning, artificial intelligence, natural language processing and other fields, so that Python is an excellent programming language that is now widely regarded.

NLTK (Natural Language Toolkit) is one of the most widely used Python libraries in Natural Language processing. NLTK is a Python library that can process Natural Language text quickly and easily. The toolkit was developed at the University of Pennsylvania as a research and teaching tool for natural language processing. NLTK has a large number of built-in corpora, including various types of text materials such as novels, news, network chat texts, film reviews, etc. It includes Brown's Corpus, Gutenberg's Corpus, Inaugural Address Corpus, Reuters Corpus, etc. (Kambhampati, 2019). In addition, NLTK provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum. In collaboration with Python's powerful standard library and other third-party libraries, it can conduct secondary processing of the processing results. It provides a solid backing for processing complex text (Li, 2019).

The NLTK library is not a Python standard library, so it needs to be downloaded before using it. The platform used in this paper is Windows, the Python version is 3.6.2, and the download and installation of NLTK is completed by using the PIP tool. Use "pip install nltk" in command line to complete the automatic download and installation, as shown in Figure 1.

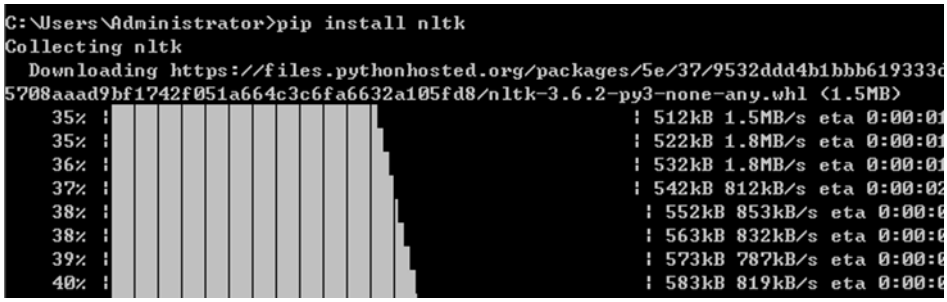

Figure 1. NLTK download and installation

After NLTK is installed, the necessary datasets / models need to be installed to use specific functions. You can install the packages by running the following code in Python's Interactive Development Environment (IDLE):

>>>import nltk

>>>nltk.download()

This will open the graphical NLTK Downloader, in which you can download various corpora, models, etc. as shown in Figure 2.

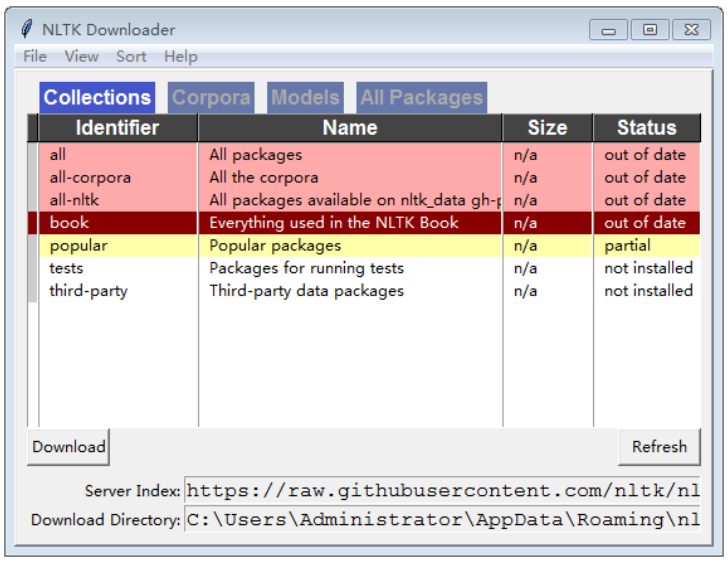

Figure 2. NLTK downloader

You can select the collection you want in the "Collections" tab. It is recommended that you select "all" to install all

the collections. If you need a corpus related to books, you can select “book” and then click the “Download” button to download. The corpus such as Moby Dick, Sense and Sensibility, The book of Genesis, Inaugural Address Corpus and so on can be available. If you only want to download the corpus you need, you can switch to the “Corpora” tab, and then select the corresponding corpus to download, such as “inaugural”.

If you only need the Inaugural Address Corpus, you can also download it with the following code in IDLE:

```
>>>nltk.download('inaugural')
```

III. Application Cases

The inaugural address of the president of the United States is publicly delivered by the President-Elect on the inauguration day, which comprehensively reflects the basic policies and guidelines of the new president in politics, economy, foreign affairs, military and other aspects. Every speech was written by excellent speech writers. From the vocabulary, syntactic structure, rhetorical devices and so on, every speech is a masterpiece. Therefore, speech has become a hot topic in the field of linguistics. The Inaugural Address Corpus in NLTK contained presidential Inaugural speeches from 1789 to 2017, with 59 texts and a total number of 149,797 words. The corpus is distinguished according to the age of speech, which corresponds to independent subtexts. This paper takes Inaugural Address Corpus as the object of English study to introduce the use of NLTK library in natural language processing.

A. Corpus Import and Show

The Inaugural Address Corpus used in this case is derived from the NLTK library and the installation method has been described earlier, it can be imported with the command “import” before use. Each corpus contains many files or documents. To get a list of these files, you can use the corpus’s fields() method. The result of viewing the Inaugural Address Corpus is shown in Figure 3. The code is as follows:

```
>>>import nltk
>>>from nltk.corpus import inaugural          # Import the Inaugural Address Corpus
>>>print (inaugural.fileids())                # Output the corpus file name
```

```
>>> print(inaugural.fileids())
['1789-Washington.txt', '1793-Washington.txt', '1797-Adams.txt',
'1801-Jefferson.txt', '1805-Jefferson.txt', '1809-Madison.txt', '
1813-Madison.txt', '1817-Monroe.txt', '1821-Monroe.txt', '1825-Ad
ams.txt', '1829-Jackson.txt', '1833-Jackson.txt', '1837-VanBuren.
txt', '1841-Harrison.txt', '1845-Polk.txt', '1849-Taylor.txt', '1
853-Pierce.txt', '1857-Buchanan.txt', '1861-Lincoln.txt', '1865-L
incoln.txt', '1869-Grant.txt', '1873-Grant.txt', '1877-Hayes.txt'
, '1881-Garfield.txt', '1885-Cleveland.txt', '1889-Harrison.txt',
'1893-Cleveland.txt', '1897-McKinley.txt', '1901-McKinley.txt', '
1905-Roosevelt.txt', '1909-Taft.txt', '1913-Wilson.txt', '1917-Wi
lson.txt', '1921-Harding.txt', '1925-Coolidge.txt', '1929-Hoover.
txt', '1933-Roosevelt.txt', '1937-Roosevelt.txt', '1941-Roosevelt
.txt', '1945-Roosevelt.txt', '1949-Truman.txt', '1953-Eisenhower.
txt', '1957-Eisenhower.txt', '1961-Kennedy.txt', '1965-Johnson.tx
t', '1969-Nixon.txt', '1973-Nixon.txt', '1977-Carter.txt', '1981-
Reagan.txt', '1985-Reagan.txt', '1989-Bush.txt', '1993-Clinton.tx
t', '1997-Clinton.txt', '2001-Bush.txt', '2005-Bush.txt', '2009-O
bama.txt', '2013-Obama.txt', '2017-Trump.txt']
```

Figure 3. List of documents in Inaugural Address Corpus

B. Preprocessing of Original Corpus

The original corpus can be obtained by web crawling, manual input or software recognition and conversion, and this case corpus is from NLTK. Usually these texts are unstructured, there are many problems of nonstandard format in the text, such as punctuation in Chinese and English, case-sensitive, special symbol, useless space and so on. Therefore, before the corpus research, we need to do some pre-processing for the text to solve these nonstandard problems. In Python, you can call some methods of string to clean up the text. Such as isalpha() method, it can determine whether the symbols in the text are letters, so as to filter out the non-letter parts of the text. The lower() method can convert uppercase letters to lowercase letters, so that words in the text can be lowercase processed. The strip() method can remove the spaces around the string, thus removing the useless spaces before and after the word.

After finishing the first step of text cleaning, we can take the next step called tokenization which is to cut the string in the text into a list of recognizable words. In this case, we directly call the words() method of inaugural corpus to get the word list of text. In addition, we can also use the word_tokenzie() method to achieve word tokenization.

In order to have more accurate data in the next step, we need to further clean and filter the tokenization results. The text contains some stop words, such as “is”, “be”, “to”, “a” and so on. These words are meaningless for research, so they should be deleted. This work can be done by calling the stopwords corpus in NLTK, which contain the common high-frequency words with no practical meaning (Li, 2019). In this case, importing the English stopwords corpus of NLTK, and filtering out the words belonging to the stopwords corpus in the Inaugural Address Corpus.

The specific code of cleaning text is as follows:

```
>>>import nltk
>>>from nltk.corpus import inaugural                    # Import the Inaugural Address Corpus
>>>from nltk.corpus import stopwords                    # Import the stop word corpus
>>>text = nltk.Text(inaugural.words())                  # Read the Inaugural Address Corpus
>>>text = [ word.lower() for word in text if word.isalpha() ]   # Convert letters in words to lowercase
>>>print(len(text))                                     # Output the number of words
>>>T = len(set(text)) / len(text)                       # Calculate word richness
>>>print(T)                                             # Output word richness
>>>stop_words = set( stopwords.words( 'english' ) )     # Import English stop words
>>> filtered_words = [ word for word in text if word not in stop_words ]
                                  # Extract words from Inaugural Address Corpus that are not in the Stop Words Corpus
```

In the above code, after importing the corpus, the isalpha() and lower() method are used to preserve and unify the English words in the text into lowercase, and then the list of words is filtered by using the stopwords corpus. The built-in Python function len() is used to calculate the number of imported words, and the set() function is used to eliminate duplicate words in the text to calculate the word richness T. Vocabulary richness is used to analyze the number of words in the text and reflect the overall use of words in the text. The lexical richness of the Inaugural Address Corpus is 6.713%. The larger the T value is, the richer the vocabulary in the text, and the use of the vocabulary in the text can be visually displayed numerically.

C. Part of Speech Tagging and Lemmatization

The English word has different forms, such as singular form, plural form, tense, and so on. For example, “do” has five forms: “do”, “dose”, “did”, “done” and “doing”. In the actual study of the word, different forms of the same word need to be combined as if they were the same word. This process is called lemmatization. The purpose of lemmatization is to restore different forms of words to a common basic form. If we do not lemmatize the form of words, there will be a big deviation in the statistical results. Lemmatization can be achieved by using WordNetLemmatizer module provided by NLTK (Deng, 2017). The lemmatize() method is used to lemmatize the word form, whose first parameter is the word, and the second parameter is the part of speech of the word, such as noun, verb, adjective, etc. The returned result of the method is the result of lemmatization of the input word.

So the part of speech of each word needs to be determined before lemmatization, so as to get the accurate result. Part of Speech Tagging is the process of automatically marking the parts of speech of all words in text according to the context information in the text. That is, corresponding labels are added after all kinds of nouns, adjectives and verbs in the text to facilitate retrieval and processing (Liu, 2015). The Part of Speech Tagging can be achieved by calling the pos_tag() method in NLTK. The specific code is as follows:

```
>>>from nltk import pos_tag                            # Import pos_tag
>>>from nltk.corpus import wordnet                      # Import Semantic Dictionary wordnet
>>>from nltk.stem import WordNetLemmatizer              # Import lemmatization tool
>>>wnl = WordNetLemmatizer()
>>>words_tag = pos_tag( filtered_words )                # Add part of speech tags
>>>original_words = [ wnl.lemmatize(i, j[0].lower()) if j[0].lower() in [ 'a' , 'n' , 'v' ] else wnl.lemmatize(i) for i, j in
words_tag ]                                             #Complete lemmatization according to part of speech
>>>words_tag [:30]                                     # PoS tagging results
>>>original_words [:30]                                #lemmatization results
```

In the above code, import the necessary libraries and modules first, and then calling the pos_tag() method for Part of Speech Tagging. Finally, lemmatize() method is used to lemmatize all the words in the word list.

Part of Speech Tagging can not only accurately restore the form of words, but also help to analyze the sentence components and divide the sentence structure. Part of Speech Tagging adds a part of speech tag to each term, as shown in Figure 4. For example, "fellow" is marked as an adjective, "citizens" is marked as a plural noun, "among" is marked as a preposition.

```
>>> words_tag[:30]
[('fellow', 'JJ'), ('citizens', 'NNS'), ('senate', 'VBP'), ('hous
e', 'NN'), ('representatives', 'NNS'), ('among', 'IN'), ('vicissi
tudes', 'NNS'), ('incident', 'JJ'), ('life', 'NN'), ('event', 'NN
'), ('could', 'MD'), ('filled', 'VB'), ('greater', 'JJR'), ('anxi
eties', 'NNS'), ('notification', 'NN'), ('transmitted', 'VBD'), (
'order', 'NN'), ('received', 'VBN'), ('day', 'NN'), ('present', '
JJ'), ('month', 'NN'), ('one', 'CD'), ('hand', 'NN'), ('summoned'
, 'VBD'), ('country', 'NN'), ('whose', 'WP$'), ('voice', 'NN'), (
'never', 'RB'), ('hear', 'JJ'), ('veneration', 'NN')]
```

Fig 4. Results of the first 30 parts of speech tagging

The lemmatization results are shown in Figure 5. Through the comparison, we can see that the words such as "citizens", "filled", "forms", "flags" are successfully restored to "citizen", "fill", "anxiety" and "transmit".

```
>>> original_words[:30]
['fellow', 'citizen', 'senate', 'house', 'representative', 'among
', 'vicissitude', 'incident', 'life', 'event', 'could', 'fill', '
greater', 'anxiety', 'notification', 'transmit', 'order', 'receiv
e', 'day', 'present', 'month', 'one', 'hand', 'summon', 'country'
, 'whose', 'voice', 'never', 'hear', 'veneration']
```

Fig 5. Lemmatization results of the first 30 words

D. Analysis and Statistics

After text cleaning, Part of Speech Tagging, lemmatization and other processing, the text can basically meet the needs of research and it can be used for vocabulary, sentence, text and other levels of analysis and research. The operations for a single word include context extraction, words in the same context extraction, double conjunctions extraction and so on. Part of Speech Tagging and syntactic analysis can be used for a single sentence. Text analysis and statistical analysis can be carried out in text, among which statistical analysis is the most commonly used tool (Wiebke, 2010).

NLTK provides a large number of tools for conducting these studies, and only several commonly used tools are described in this article.

NLTK provides three methods for the context retrieval of the target word (Liu, 2019), using concordance() to retrieve the output of the sentence containing the target word, using common_contexts() to find the common context of the vocabulary set, and using similar() to find words that have similar meaning and usage to the specified word. Through the above three methods, we can find the target vocabulary and provide the basis for the next step of analysis The sample code is shown below, and the result is shown in Figure 6.

```
>>>text = nltk.Text( inaugural.words() )
>>>text.concordance( 'China' )              # Search the text for the frequency the word "China" appears and its context
>>>text.common_contexts( ['this', 'that'] ) # Search text for words that are common in the context of “this” and “that”
>>>text.similar( 'country' )                # Search text for similar words that appear in the context of “country”
```

```
>>> text.concordance('China')
Displaying 2 of 2 matches:
honorably in the thrilling scenes in China , while new to American life , has b
he French soldier who dies in Indo - China , the British soldier killed in Mala
>>> text.common_contexts(['this','that'])
of_country of_government of_day and_is to_end but_is of_union
at_moment in_great of_character and_i from_day all_we to_high
of_conflict of_spirit of_people at_point themselves_is people_is
>>> text.similar('country')
government nation people union world time constitution land citizens
peace states laws spirit system power way faith future executive earth
```

Fig 6. Retrieval results

As the most commonly used mathematical analysis method in NLTK, probability statistics is used for data processing and analysis in text. In Python, we use the function that calculates the frequency defined in NLTK to count the word frequency, word length and other related operations on words, collocations, common expressions or symbols that appear in text. The FreqDist() in NLTK can realize the function of word frequency statistics. First call the function to create a frequency distribution, and then you can call the most_Common(n) method to extract high-frequency words from frequency distribution, and the tabulate(n) method can be called to output results in the form of table, where parameter n is the number of extracted words (Steven, 2009). Some of the results are shown in Figure 7. And the specific code is as follows:

```
>>>fdist = nltk.FreqDist( filtered_words )    # Create a frequency distribution for the cleaned and filtered words
>>>fdist.most_common(30)                      # Extract the top 30 high-frequency words
>>>fdist ['target word']
                    # Use the created frequency distribution to find the number of occurrences of the target vocabulary
>>>fdist.tabulate()                           # Output in tabular form
```

```
>>> fdist.most_common(30)
[('government', 600), ('people', 584), ('us', 478), ('upon',
371), ('must', 366), ('great', 340), ('may', 338), ('world',
338), ('states', 333), ('country', 318), ('nation', 316), ('
shall', 314), ('every', 299), ('one', 257), ('peace', 255),
('new', 252), ('citizens', 247), ('power', 236), ('public',
226), ('america', 220), ('time', 217), ('would', 211), ('con
stitution', 206), ('united', 202), ('nations', 199), ('union
', 189), ('freedom', 189), ('free', 184), ('war', 178), ('am
erican', 163)]
```

Figure 7. The top 30 items with the highest frequency

E. Graphic Display

In addition to the direct output of data results, you can also use Python's third-party libraries for secondary processing of data to display data results in a visual form, which is more intuitive and better.

Matplotlib is a library for visualization in Python, which can be used to draw statistical charts for structured data, such as histogram, sector chart, line chart, histogram and so on. The statistical data obtained in the previous stage are displayed in the form of frequency line chart, as shown in Figure 8, from which we can see the comparison of different vocabulary frequencies. The code is as follows:

```
>>>fdist.plot(30)                    # The top 30 items with the highest frequency are shown in line charts
```

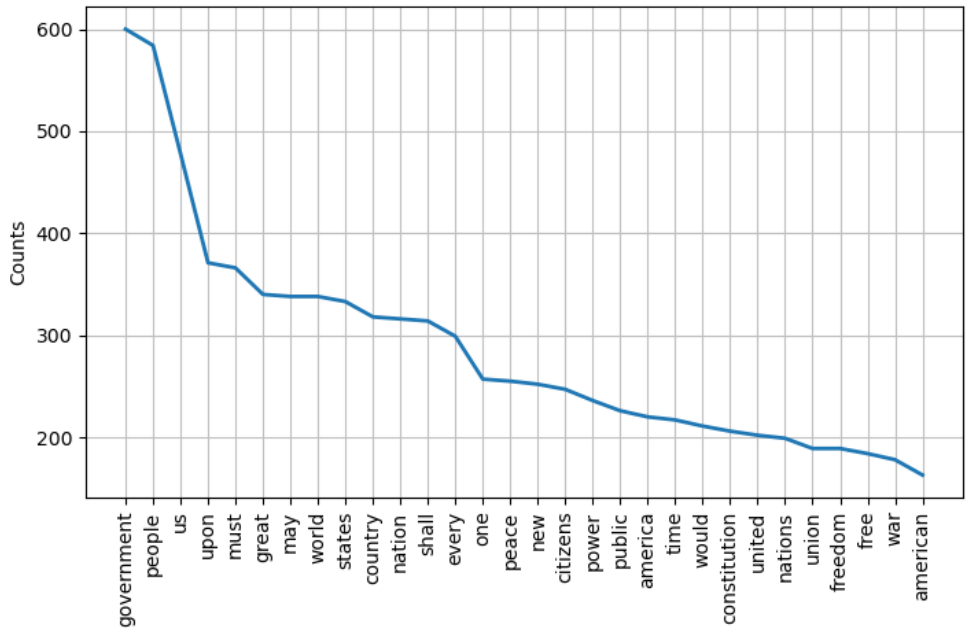

Fig 8. Line chart of word frequency

You can also use the dispersion_plot() method to show the positions of words in the text in the form of a discrete graph. The implementation code is shown below, and the results are shown in Figure 9. It can be seen from the figure that the frequency of "us" in recent years is significantly higher than that in the early stage. Combined with the chronological structure of the inaugural address corpus, we can see that there are significant differences in the frequency of different speech words over time.

```
>>>text.dispersion_plot( ['government', 'people', 'us', 'citizens'] )
        # Show the position of "government", "people", "us" and "citizens" in the text in the form of discrete graph
```

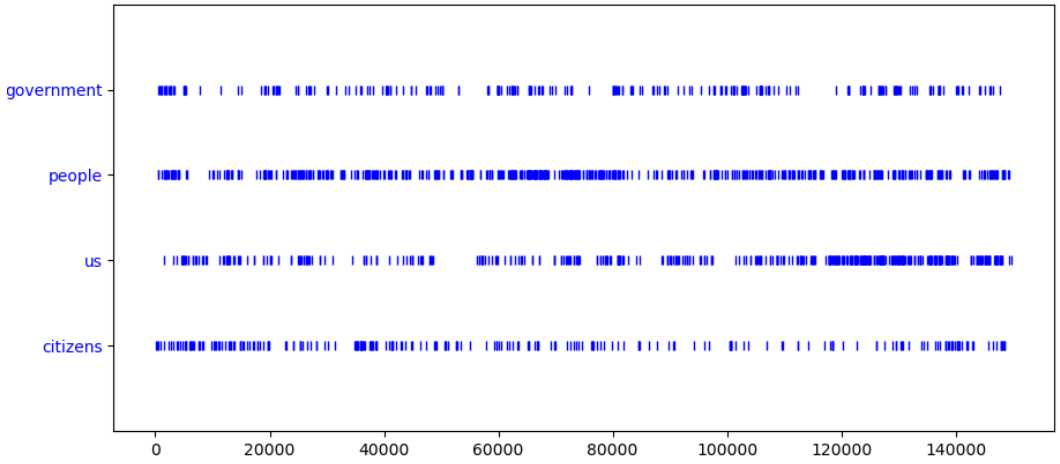

Fig 9. Discrete graph

An interesting feature of the inaugural addresses corpus is its temporal dimension, so we can compare the frequency of keywords used in inaugural addresses in different years to see the usage of words over time. We can use the NLTK ConditionalFreqDist() method to see how many times each keyword appeared in speeches over time. The following code takes the two keywords "American" and "citizen" as an example. First, we use w. lower() to convert the words in the inaugural addresses corpus into lowercase, then use startswith() to check whether they start with the target words "American" and "citizen", and finally count the frequency of words in each speech text. The statistical results are displayed in the form of line chart, as shown in Figure 10. It can be seen from the figure that the word "citizen" peaked in the text of 1941. The specific implementation code is as follows:

```
>>> cfd = nltk.ConditionalFreqDist( ( target, fileid[:4] )
                                    for fileid in inaugural.fileids()
                                    for w in inaugural.words( fileid )
                                    for target in [ 'american', 'citizen' ]
                                    if w.lower().startswith( target ))
                                                #Use the conditional probability distribution method ConditionalFreqDist
>>> cfd.plot()                                 # The frequency of two keywords is compared with the line chart
```

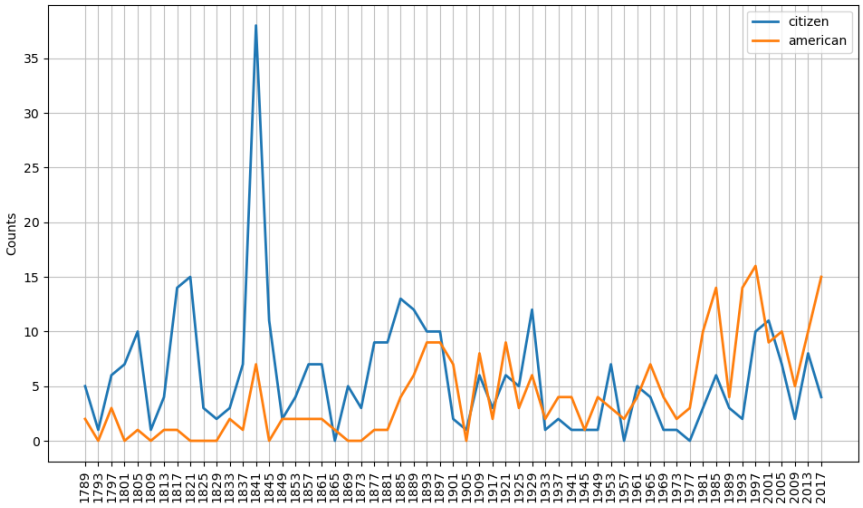

Fig 10. Line chart of key words frequency

In addition, the word cloud image is also an effective form of data display. Word cloud image, also known as text cloud, is a visual display of high frequency words in the text. Word cloud image filters out a large number of low-frequency and low-quality text information, so that the viewer can appreciate the theme of the text as long as he has a glance. The wordcloud library in Python can be used to generate all kinds of beautiful word cloud image. In this case, the results of lemmatization are displayed in word cloud. First, import the wordcloud library and Matplotlib library, then call the generate() function of the WordCloud module to generate the word cloud. Finally, use the pyplot function in Matplotlib library to display the word cloud. The word cloud image is shown in Figure 11, and the specific code is as follows:

```
>>>from wordcloud import WordCloud                    # Import WordCloud
>>>import matplotlib.pyplot as plt                    # Import pyplot
```

```
>>>cut_text = " ".join(original_words)
                    # Concatenates the words in the original_words list into a string separated by spaces
>>>wordcloud = WordCloud( font_path = "C:/Windows/Fonts/Cambria.ttf", background_color = "white", width = 1000, height = 880 ).generate( cut_text )
                    # Set the font, background color, width and height of the image, then generate word cloud
>>>plt.imshow(wordcloud, interpolation = "bilinear")    # Draw the image
>>>plt.axis("off")                                      # Don't show axis
>>>plt.show()                                           # Display the word cloud
```

Fig 11. The word cloud image

IV. Conclusion

NLTK, based on Python language, supports a large number of corpora and it is widely applied in natural language processing, computational linguistics, scientific computational analysis and other aspects with powerful functions. At present, in the domestic research based on the corpus, the commonly used tools are WordSmith, Antconc, Range, etc., while NLTK of Python is rarely used for research. The reason for this is that most researchers do not have the Python programming skills to take full advantage of NLTK's capabilities. Aiming at the main steps in corpus research and taking the US presidential inaugural address corpus as an example, this paper introduces how to use the NLTK of Python to process the corpus, and how to use the third-party Libraries of Python to data visualization, so that corpus researchers can be familiar with and use it, enrich their research tools, promote the development of corpus linguistics research, and promote the cross application of computer in different disciplines.

References

[1] Deng Qingqiong, Peng Weiming, Yin Gan. (2017). A case study of practical word frequency statistics in Python teaching. *Computer Education* 12, 20-27.

[2] Feng Min. (2020). Research on corpora in College English Grammar Teaching. *China Journal of Multimedia & Network Teaching* 11, 150-152.

[3] Kambhampati Kalyana Kameswari, J Raghaveni, R. Shiva Shankar, Ch. Someswara Rao. (2019). Predicting Election Results using NLTK. *International Journal of Innovative Technology and Exploring Engineering* 9.1, 4519-4529.

[4] Li Junfei. (2019). Research on the Application of Natural Language Processing Toolkit in College English Teaching. *Education Modernization* 92, 136-137.

[5] Li Chen, Liu Weiguo. (2019). Chinese Text Information Extraction Based on NLTK. *Computer Systems & Applications* 28.1, 275−278.

[6] Liu Xu. (2015). The Application of NLTK Toolkit Based on Python in Corpus Research. *Journal of Kunming Metallurgy College* 31.5, 65-69.

[7] Liu Weiguo, Li Chen. (2019). Case design of NLTK module application in Python programming teaching. *Computer Education* 3, 92-97.

[8] Steven B, Ewan K, Edward L. (2009). NLTK Natural Language Processing with Python. California: O 'Reilly Media.

[9] Wiebke Wagner. (2010). Steven Bird, Ewan Klein and Edward Loper: Natural Language Processing with Python, Analyzing Text with the Natural Language Toolkit. *Language Resources and Evaluation* 44.4, 421-424.

Meng Wang received the B.S. and M.S. degree in computer science from Shandong Normal University, China in 2003 and 2006.

He is an associate professor in the School of Medical Information Engineering, Jining Medical University, Rizhao, Shandong, China. His current research interests include computer application technology and the application of Python in English Research.

Fanghui Hu received her Master Degree from Hunan University in 2007.

She is currently a lecturer in School of Foreign Languages, Jining Medical University, Rizhao, Shandong, China. She has been teaching English courses for more than ten years. Her research interests include second language acquisition and language testing.